ORIENTAL JOURNAL OF
COMPUTER SCIENCE & TECHNOLOGY
An International Open Free Access, Peer Reviewed Research Journal
Published By: **Oriental Scientific Publishing Co., India.**
www.computerscijournal.org

ISSN: 0974-6471
December 2015,
Vol. 8, No. (3):
Pgs. 234-240

Scheduling Strategies in Hadoop: A Survey

S. RASHMI[1] and ANIRBAN BASU[2]

[1]Department of Computer Science and Engineering,
East Point College of Engineering and Technology, Bengaluru-560049, India.
[2]Department of Computer Science and Engineering,
APS College of Engineering, Bengaluru-560082, India.

(Received: November 10, 2015; Accepted: December 09, 2015)

ABSTRACT

Hadoop-MapReduce is one of the dominant parallel data processing tool designed for large scale data-intensive cloud computing platforms. Hadoop Mapreduce framework requires a proficient mechanism to allocate and schedule computing resources which makes it liable to accomplish the desired preferences for the specific business application. In this paper, an analysis of a few scheduler improvement methods that improves the completion time of the job or the task has been carried out in order to facilitate a judicious choice of the scheduler based on the requirement and application expectations .

Key words: Cloud Computing, Hadoop, MapReduce, Scheduling.

INTRODUCTION

Cloud computing[1] is an emerging technology which enables convenient, on-demand access to a shared pool of configurable computing resources, such as servers, networking, applications and services. These shared resources can be rapidly deployed and re-deployed with minimal human intervention to meet resource requirements. The resources can be rapidly changed to match the immediate needs of the workload in minutes instead of hours or days. For example, multiple computer-aided engineering loads can process faster in an environment that is able to scale to meet demand, which makes cloud computing efficient, flexible and collaborative. Among the many applications which benefit from cloud and cloud technologies, the data/compute intensive applications are the most important. The overflow of data and the highly compute intensive applications found in many domains such as particle physics, biology, chemistry, finance, and information retrieval, mandate the use of large computing infrastructures and parallel processing to achieve considerable performance gains in analyzing data.

Cloud technologies[2] create new trends in performing parallel computing. Hadoop - MapReduce has grown to be one among the prominent parallel data processing tool designed for large scale data-intensive cloud computing platforms. MapReduce automatically parallelizes computation[3] by running multiple map and/or reduce tasks over distributed data across multiple machines.

Scheduling is a methodology of allocating system resources like CPU time, memory, bandwidth etc., to various jobs, so that the efficient load balancing takes place and the performance of the system improves. Efficient resource allocation and management in data centers and clouds running large distributed data processing frameworks like MapReduce is crucial for enhancing the performance of hosted applications and to the desired levels of high availability and faster service. There are default schedulers like FIFO, Fair Scheduler, Capacity Scheduler[4-8], but they come with their own merits and demerits.

Many algorithms have been applied to solve the stated problem. Here, a study of scheduling methods has been made, and the different situations where one performs better than others.

Background information

Hadoop-MapReduce

Hadoop [9] is an open-source software framework for storage and large-scale processing of data-sets on clusters of commodity hardware. A small Hadoop cluster includes a single master and multiple worker nodes .

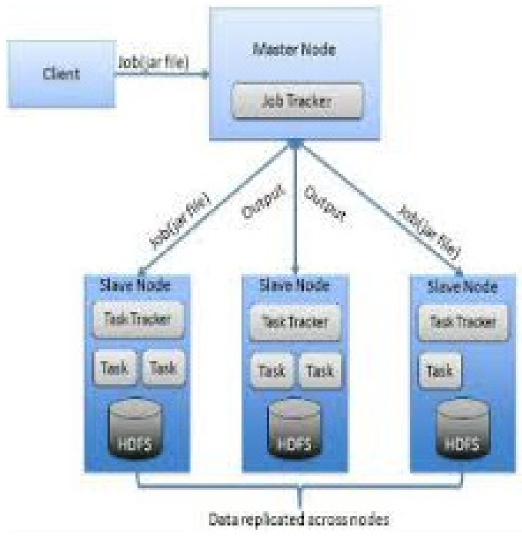


Fig. 1: Hadoop - MapReduce Architecture

The *master* node consists of a JobTracker and a NameNode. A *slave* or *worker node* acts as both a DataNode and TaskTracker. In a larger cluster, the HDFS[10] is managed through a dedicated NameNode server to host the file system index, and a secondary NameNode that can generate snapshots of the namenode's memory structures, thus preventing file-system corruption and reducing loss of data.

The key objective of Map-Reduce programming model is to parallelize the job execution across multiple nodes for execution. It creates multiple tasks to be executed and executes them on the multiple machines. The large Map-Reduce cluster is used to execute multiple jobs of different users multiple combinations of task and machine are possible. Figure 1 shows Hadoop-MapReduce architecture.

Primary hadoop schedulers

MapReduce in Hadoop comes with a choice of schedulers.

- FIFO Scheduler: The default FIFO [4][5] is a queue-based scheduler . Smaller jobs arrived subsequently may be deferred until the initial large jobs complete.
- Fair Scheduler: Fair scheduler [5] strives to maintain a sense of balance between all jobs arriving by equally distributing the readily available resources.
- Capacity Scheduler: Capacity scheduler[6] assures a dedicated queue so that each user can commence the moment it is arrives, thereby maximizing throughput and utilization in cluster.

Other improved scheduling methods

Resource Stealing

Guo et al present a Resource Stealing[11] mechanism which can be intermixed with any of the prevailing job schedulers such as Fair scheduler or Capacity Scheduler to improve Resource Utilization in a Heterogeneous Environment. Residual resources are idle slots which may remain wasted if not utilized. Resource stealing takes advantage of this information and allows the executing tasks to steal the residual resources which speeds up its execution as supplementary resources are now added .These stolen resources may be restored back by the node whenever it starts a new task. There are a number of policies which describes how to share the residual resources among the current running tasks. Table 1 summarizes the Allocation policies.

COSHH: COSHH[12] is a Classification and Optimization based scheduler for Heterogeneous Hadoop Systems

The objectives of COSHH shown in Fig 2. The first two objectives is taken care of in many schedulers but they assume the ideal case of homogeneity in the system. It tries to meet the first objectives by reducing the scope of search by proper classification into job classes. Any arrival of a job needs to confine its search only with the job class that shares identical properties like priority, mean execution rate and mean arrival rate with it. It defines and solves a Linear Program (LP) to get a set of suggested classes. It also uses a secondary optimization and classification that perks up Locality and in turn handles Communication and Search overhead. COSHH has an improved average completion time even the system workload varies.

PRISM: Fine-Grained Resource-Aware Scheduling for MapReduce

PRISM[13] is a method of Resource Allocation which tries to improve resource utilization and job running time considering Phase- Level Scheduling. A job is a collection of multiple tasks. A MapReduce job mainly consists of Map and Reduce tasks but the programming model has multiple phases like

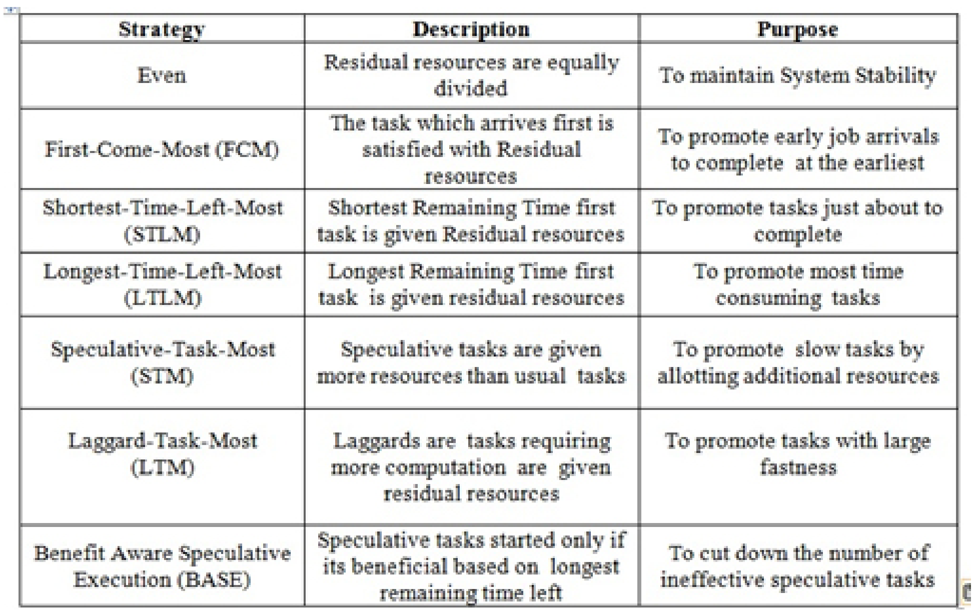
1. Map
2. Merge
3. Shuffle
4. Sort
5. Reduce

PRISM, a Phase and Resource Information-aware Scheduler for MapReduce clusters emphasizes on the fact that resource requirements not only varies at Job Level or Task Level but also varies at Phase Level. For any application, the resource requirements may be as follows:

- Map Phase – High CPU , Low Storage , I/O
- Merge Phase – High I/O , Storage
- Shuffle Phase – High Bandwidth, Low CPU
- Sort Phase – High CPU
- Reduce Phase – Low CPU,I/O

Qi Zhang et al define a scheduling method that consists of three elements: a Phase –based Scheduler, Local Node manager and a Job

Table 1: Allocation Policies Of Different Residual Resources



Strategy	Description	Purpose
Even	Residual resources are equally divided	To maintain System Stability
First-Come-Most (FCM)	The task which arrives first is satisfied with Residual resources	To promote early job arrivals to complete at the earliest
Shortest-Time-Left-Most (STLM)	Shortest Remaining Time first task is given Residual resources	To promote tasks just about to complete
Longest-Time-Left-Most (LTLM)	Longest Remaining Time first task is given residual resources	To promote most time consuming tasks
Speculative-Task-Most (STM)	Speculative tasks are given more resources than usual tasks	To promote slow tasks by allotting additional resources
Laggard-Task-Most (LTM)	Laggards are tasks requiring more computation are given residual resources	To promote tasks with large fastness
Benefit Aware Speculative Execution (BASE)	Speculative tasks started only if its beneficial based on longest remaining time left	To cut down the number of ineffective speculative tasks

Progress Monitor. The Phase –based Scheduler depends on Job Profilers such as Starfish[14] to gather information about resource requirements. The Node manager synchronizes the progress of task at Phase level excluding Stragglers.

Hadoop Scheduling Base on Data Locality

Bo et al present a scheduling method[15] which address the problem of data locality. Hadoop-MapReduce supports fault tolerance by keeping three copies of data in different nodes. Data required for computation may be in the same node, in a different node but in the same rack or in a remote node. The scheduler has three main sections:

1. Node –Preselecting
2. Task – Preselecting
3. Resource Prefetch

In the first section , it performs a node pre-select based on either of the better, remaining execution time or data transmission time. The second section is a task preselects which chooses a local task, if one exists or chooses a remote task. Last and important section is the Resource Prefetch .It determines the source node and the destination node and calculates a nearest point. It then fetches resources to local nodes reducing communication cost causing less overhead. Hence, the scheduler handles not just the data locality problem but also improves resource utilization and job response time.

Table 2: Comparison of a few Schedulers

Algorithm	Main Feature	Dynamic	Heterogeneity	Data Locality
FIFO	FIFO	N	N	N
Fair	Fairness	Y	N	N
Capacity	Priority	Y	N	Y
BASE	Speculation	Y	Y	N
COSHH	Optimization	Y	Y	N
DREAMS	Data Skewness	Y	N	N
PRISM	Phase-Level Scheduling	Y	N	N
Hadoop Scheduling Base	Data Locality	Y	N	Y
Dynamic Workload Balancing	Rack Level	Y	Y	N
LBNP	Node Performance	Y	Y	N

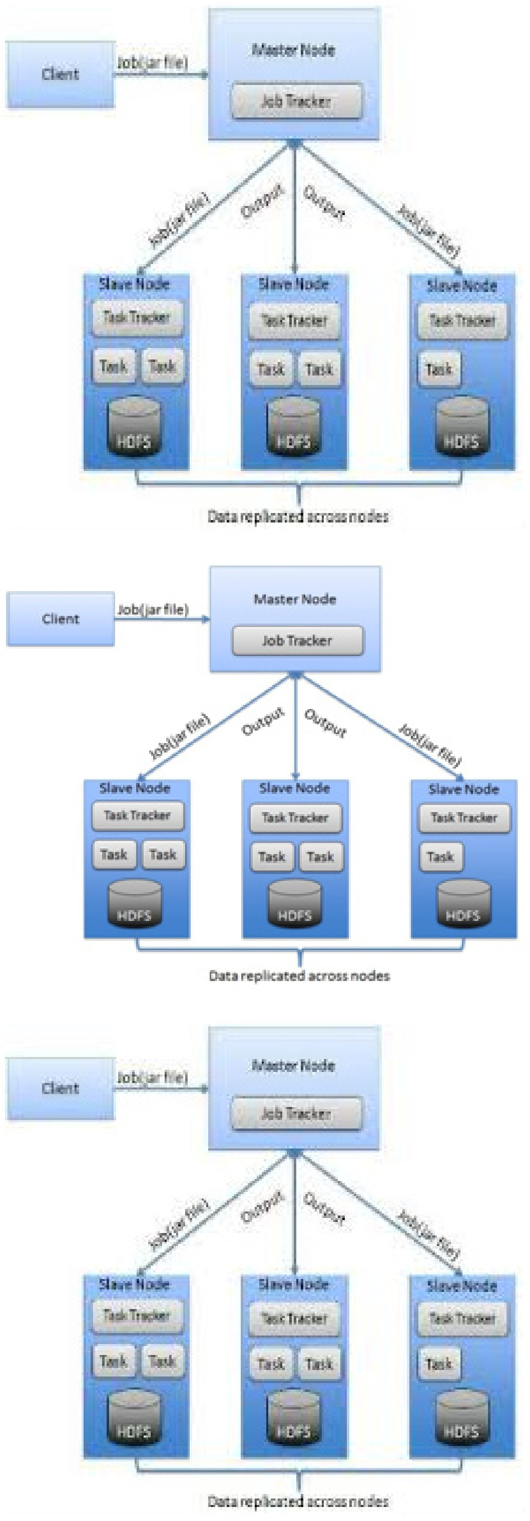
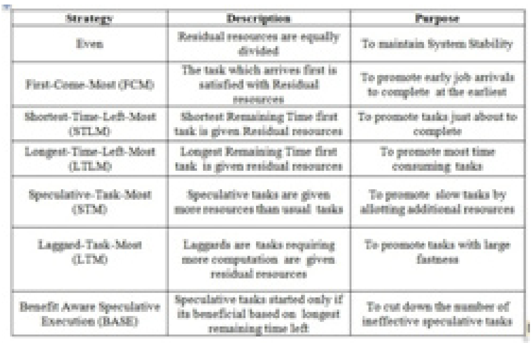


Fig. 2: Hadoop - MapReduce Architecture

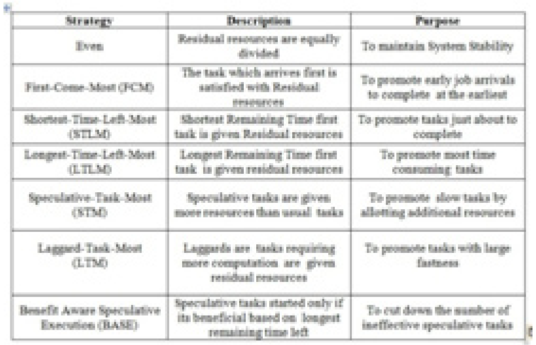
DREAMS: Dynamic Resource Allocation for MapReduce with Data Skew

Applications in the field of Microbiology, Psychology, Web Technology etc., consists of enormous size of skewed data. Data skewness is a crucial factor, if not handled cleverly, may not effectively utilize the resource and deteriorate performance. Liu et al define a DREAMS[16] framework which aims to shrink the negative influence of skewed on performance of the system. DREAMS consist of a Prediction model which dynamically projects the partition the partition size of reduce tasks which is highly influenced by past data of the application. A Progressive Sampling [17] method may be used for the purpose. DREAMS then constructs a performance model by discovering the affect of Task duration on CPU allocation, Partition Size and Memory Allocation. This data assists

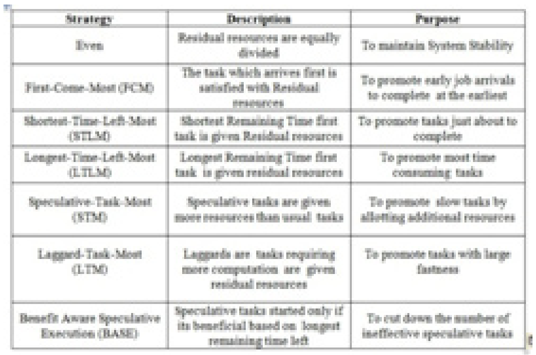
Table 3: Allocation Policies Of Different Residual Resources



Strategy	Description	Purpose
Even	Residual resources are equally divided	To maintain System Stability
First-Come-Most (FCM)	The task which arrives first is satisfied with Residual resources	To promote early job arrivals to complete at the earliest
Shortest-Time-Left-Most (STLM)	Shortest Remaining Time first task is given Residual resources	To promote tasks just about to complete
Longest-Time-Left-Most (LTLM)	Longest Remaining Time first task is given residual resources	To promote most time consuming tasks
Speculative-Task-Most (STM)	Speculative tasks are given more resources than usual tasks	To promote slow tasks by allotting additional resources
Laggard-Task-Most (LTM)	Laggards are tasks requiring more computation are given residual resources	To promote tasks with large fastness
Benefit Aware Speculative Execution (BASE)	Speculative tasks started only if its beneficial based on longest remaining time left	To cut down the number of ineffective speculative tasks

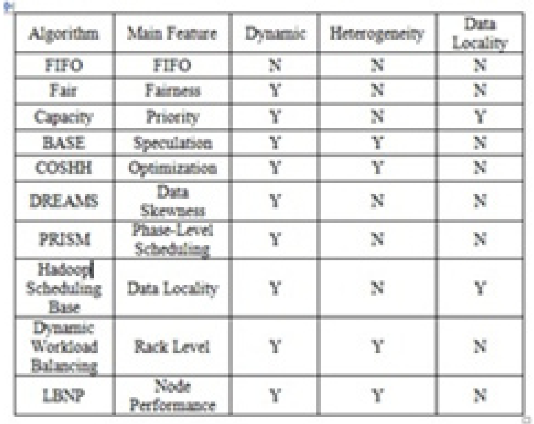


Strategy	Description	Purpose
Even	Residual resources are equally divided	To maintain System Stability
First-Come-Most (FCM)	The task which arrives first is satisfied with Residual resources	To promote early job arrivals to complete at the earliest
Shortest-Time-Left-Most (STLM)	Shortest Remaining Time first task is given Residual resources	To promote tasks just about to complete
Longest-Time-Left-Most (LTLM)	Longest Remaining Time first task is given residual resources	To promote most time consuming tasks
Speculative-Task-Most (STM)	Speculative tasks are given more resources than usual tasks	To promote slow tasks by allotting additional resources
Laggard-Task-Most (LTM)	Laggards are tasks requiring more computation are given residual resources	To promote tasks with large fastness
Benefit Aware Speculative Execution (BASE)	Speculative tasks started only if its beneficial based on longest remaining time left	To cut down the number of ineffective speculative tasks

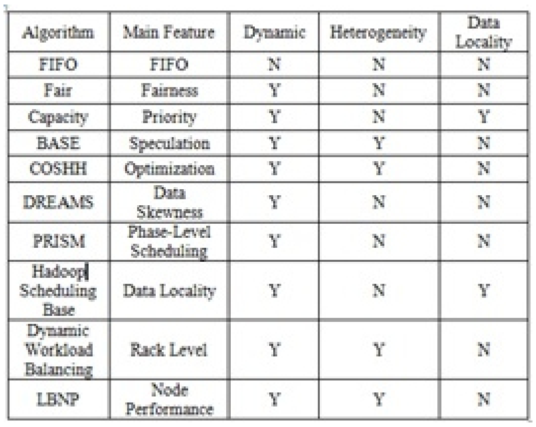


Strategy	Description	Purpose
Even	Residual resources are equally divided	To maintain System Stability
First-Come-Most (FCM)	The task which arrives first is satisfied with Residual resources	To promote early job arrivals to complete at the earliest
Shortest-Time-Left-Most (STLM)	Shortest Remaining Time first task is given Residual resources	To promote tasks just about to complete
Longest-Time-Left-Most (LTLM)	Longest Remaining Time first task is given residual resources	To promote most time consuming tasks
Speculative-Task-Most (STM)	Speculative tasks are given more resources than usual tasks	To promote slow tasks by allotting additional resources
Laggard-Task-Most (LTM)	Laggards are tasks requiring more computation are given residual resources	To promote tasks with large fastness
Benefit Aware Speculative Execution (BASE)	Speculative tasks started only if its beneficial based on longest remaining time left	To cut down the number of ineffective speculative tasks

Table 4: Comparison of a few Schedulers



Algorithm	Main Feature	Dynamic	Heterogeneity	Data Locality
FIFO	FIFO	N	N	N
Fair	Fairness	Y	N	N
Capacity	Priority	Y	N	Y
BASE	Speculation	Y	Y	N
COSHH	Optimization	Y	Y	N
DREAMS	Data Skewness	Y	N	N
PRISM	Phase-Level Scheduling	Y	N	N
Hadoop Scheduling Base	Data Locality	Y	N	Y
Dynamic Workload Balancing	Rack Level	Y	Y	N
LBNP	Node Performance	Y	Y	N



Algorithm	Main Feature	Dynamic	Heterogeneity	Data Locality
FIFO	FIFO	N	N	N
Fair	Fairness	Y	N	N
Capacity	Priority	Y	N	Y
BASE	Speculation	Y	Y	N
COSHH	Optimization	Y	Y	N
DREAMS	Data Skewness	Y	N	N
PRISM	Phase-Level Scheduling	Y	N	N
Hadoop Scheduling Base	Data Locality	Y	N	Y
Dynamic Workload Balancing	Rack Level	Y	Y	N
LBNP	Node Performance	Y	Y	N

Algorithm	Main Feature	Dynamic	Heterogeneity	Data Locality
FIFO	FIFO	N	N	N
Fair	Fairness	Y	N	N
Capacity	Priority	Y	N	Y
BASE	Speculation	Y	Y	N
COSHH	Optimization	Y	Y	N
DREAMS	Data Skewness	Y	N	N
PRISM	Phase-Level Scheduling	Y	N	N
Hadoop Scheduling Base	Data Locality	Y	N	Y
Dynamic Workload Balancing	Rack Level	Y	Y	N
LBNP	Node Performance	Y	Y	N

the scheduler to assign just the required quantity of resources to reduce tasks having comparable running time. A study has shown that DREAMS cuts down the Job Completion Time.

Dynamic Workload Balancing for Hadoop MapReduce

The Architecture of HDFS[10] is says data is placed in blocks of sizes 64MB by default. It consists of two types of nodes called NameNode and DataNode. The NameNode is the master node and the DataNode are the slave nodes. To preserve Reliability, Availability and Fault tolerance, HDFS maintains three replicas of an individual block. An HDFS cluster extends to multiple racks. The replicas are sited at these positions. One copy will be placed on a node in a local rack; the second copy will be placed on a different node in the same rack and the last copy on a different node in a different rack. Hence the communication costs in terms of disk transfer rate needs to be considered when dealing with large files and large block sizes.

X. Hou et al uses the concept of Software Defined Networking (SDN) and OpenFlow switch[18] to enhance performance in Hadoop. SDN makes the bandwidth adaptable to the changes in the workload of the rack. Openflow switch, an implementation of SDN, guides the movement of data among the racks. The scheduler[19] tries to achieve load balance at the rack level. The MapReduce programming model takes no notice of heterogeneity in datanodes, which causes imbalance in the rack level. A Hadoop cluster is a heterogeneous cluster with respect to the infrastructure of the datanodes. At any instant of time, the number of active datanodes may vary along with its hardware and software utilization. This causes imbalance in workload. The algorithm has three main steps:

1. The algorithm first examines the number of tasks running in each rack. This information can be calculated from the difference in start time and finish time maintained the algorithm in the log record. In addition it calculates four parameters: Processing capabilities of datanodes and racks and busy level of datanodes and racks.
2. Based on the calculated information, the algorithm chooses the busiest rack for scheduling.
3. The job with the longest remaining execution time is shifted from the rack selected in Step 2 to a less busy rack.

The study has shown an improvement in Job execution time.

LBNP – A Load Balance Algorithm based on Nodes Performance in Hadoop Cluster

Z. Gao[20] et al have given a Load Balancing algorithm which is designed to work in a heterogeneous environment. Hadoop scheduling by default assumes the ideal case of homogeneity which may not be practically possible. Workload imbalance occurs even though Hadoop nodes deal with same quantity of data, the reason being that cluster nodes have a diverse range of hardware capabilities.

LBNP has 4 steps

1. Data Pre-Allocation –Performance of all nodes are calculated and sorted. A nodes collection *S = {S1, S2, S3…Sn}* and data proportion *P = {p1, p2, p3…pn}* is obtained and S_i is mapped with P_i. If the number of available nodes are sufficient to allocate the reduce tasks, then the sorted nodes are equally divided into the number of reduce tasks. If less number of nodes are available, then the corresponding data ratio can be link to set of data and cluster.
2. Partition Phase - P_i is combined with Partition number.
3. Allocation of Reduce tasks – Select an appropriate S_i based on status of the TaskTracker. It depends on tasks upto $S_{i \text{ or}}$ S_{i-1} have been executed.
4. Evaluation of node performance - When all reduce tasks complete, update the performance of all nodes.

The analysis says that it reduces the execution time of a job.

CONCLUSION

The contemporary scheduler for Hadoop is based on fixed sized slots. This calls for the researchers to focus their work on designing new schedulers which comply with the different drawbacks of the existing scheduling methods. The paper outlines a few scheduling methods proposed by various researchers that concentrate on reducing the completion time of the job.

REFERENCES

1. NIST Definition of Cloud Computing v15,csrc. nist.gov/groups/SNS/cloud-computing/cloud - def-v15.doc
2. Jaliya Ekanayake , Xiaohong Qiu, Thilina Gunarathne, Scott Beason, Geoffrey Fox , " *High Performance Parallel Computing with Cloud and Cloud Technologies*", **34**: pp 20-38 (2010).
3. J Dean and Ghemawat ,"MapReduce: Simplied Data Processing on Large Clusters," Google, Inc., (2004)
4. Hadoop: The Definitive Guide, Second Edition, by Tom White, O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472
5. Hadoop's Fair Scheduler.https://hadoop. apache.org/ docs/r1.2.1/fair_scheduler
6. Hadoop's Capacity Scheduler http://hadoop. apache.org/core/docs/current/capacity_ scheduler.html
7. Jagmohan Chauhan, Dwight Makaroff and Winfried Grassmann ,Dept. of Computer Science, University of Saskatchewan, "The Impact of Capacity Scheduler Configuration Settings on MapReduce Jobs", Nov 2012
8. http://www.ibm.com/developerworks//library/ os-hadoop-scheduling
9. Apache. Hadoop. http://hadoop.apache.org
10. Hadoop Distributed File System,http:// hadoop.apache.org/hdfs
11. Zhenhua Guo, Geoffrey Fox , " Improving MapReduce Performance in Heterogeneous Network Environments and Resource Utilization", in 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, 2012
12. Aysan Rasooli , Douglas G. Down, "COSHH: A Classi_cation and Optimization based Scheduler for Heterogeneous Hadoop Systems", in July 2014
13. Qi Zhang, Mohamed Faten Zhani, Yuke Yang, Raouf Boutaba,and Bernard Wong," PRISM: Fine-Grained Resource-Aware Scheduling for MapReduce", in Jun 2015
14. H. Herodotou, H. Lim, G. Luo, N. Borisov, L. Dong, F. Cetin, and S. Babu, "Starfish: A self-tuning system for big data analytics," in *Proc. Conf. Innovative Data Syst. Res.,* pp. 261–272 (2011)
15. Bo Jiang, Jiaying Wu, Xiuyu Shi, Ruhuan Huang, "Hadoop Scheduling Base On Data Locality", 2015
16. Z Liu, Zhang, Zhani, Boutaba,Y Liu, Gong," DREAMS: Dynamic Resource Allocation for MapReduce with Data Skew", 2015
17. S.R.Ramakrishnan,G.Swart,and A. Urmanov, "Balancing reducer skew in mapreduce workloads using progressive sampling," in *Proceedings Of the Third ACM Symposium on Cloud Computing. ACM,* p. 16 (2012)
18. N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson,J. Rexford, S. Shenker, and J. Turner, "Openflow: enabling innovation in campus networks", *SIGCOMM Comput. Commun. Rev.,* **38**(2): 69–74: (2008)
19. X Hou, Ashwin Kumar T K,Johnson P Thomas, Vijay Varadharajan," Dynamic Workload Balancing for Hadoop MapReduce", *IEEE,* (2014)
20. Zhipeng Gao, Dangpeng Liu, Yang Yangÿ Jingchen Zheng,Yuwen Hao,"A Load Balance Algorithm Based on Nodes Performance in Hadoop Cluster" in 2014.